Optimization Project 2

Ambikha Maharaj (am84333), Anant Gupta (ag78593), Dipali Pandey (dp33957), Nir Rauch (nmr882)

Introduction

The objective of this project is to construct a portfolio that reflects the NASDAQ-100 index. It utilizes the component securities in the index to build an index fund which minimizes the difference between the fund and the index returns without having to replicate the whole market index, which could be an expensive endeavor due to frequent rebalancing requirements.

Data and Methodology

We use the price data of individual 100 stocks in the NASDAQ-100 as well as the price data of the whole index for the year 2019 as the training set and then test the model on the price data for the year 2020.

1. Indirect Method - To create an index fund with m stocks, we formulated an integer program that picks exactly m out of n stocks for our portfolio. This integer program took as input the correlation matrix of the stock returns, ϱ. We then solved a linear program to decide how many of each chosen stock to buy for our portfolio and finally evaluated how well our index fund does as compared to the NASDAQ-100 index both in sample and out of sample.

2. Direct Method - As opposed to the method above, rather than selecting the stocks first and then calculating their weights, here, we select the weights directly for individual stocks by formulating the problem as a MIP. We then calculate the performance of our fund against various values of m.

Analysis

Direct Method: We maximized the similarity between the n stocks in NASDAQ-100 and their representatives in the index fund. We used the binary programming to determine the choice of m stocks to be used in our fund:

$$max \sum_{i=1}^{n} \sum_{j=1}^{m} \rho_{ij} x_{ij} \qquad (1)$$

$$\text{s.t.} \sum_{j=1}^{n} y_i = m$$

$$\sum_{j=1}^{n} x_{ij} = 1 \; for \; i = 1 \; to \; n$$

$$x_{ij} \leq y_j \; for \; i = 1 \; to \; n$$

$$x_{ij}, y_j \in \{0, 1\}$$

Based on the stocks selected to be in the fund, we determined the relative weights of these stocks in the fund subject to the following constraints:

$$min \sum_{t=1}^{T} y_t \qquad (2)$$

$$\text{s.t.} \sum_{i=1}^{m} w_i = 1$$

$$y_t + \sum_{i=1}^{m} w_i r_{it} >= q_t \quad for \; t = 1 \; to \; T$$

$$y_t - \sum_{i=1}^{m} w_i r_{it} >= -q_t \quad for \; t = 1 \; to \; T$$

$$w_i >= 0$$

Python Code:

After computing the returns of all the stocks (and NASDAQ-100 Index) and their pairwise correlation using the price data, we defined helper functions to build our portfolio model. We first defined a function to optimize our objective using Gurobi and a function to select stocks based on correlation matrix:

```
def getfund (m,simmat,allstocks,l=''):
    if l =='l':
         return -1
    #Sets all correlations as coeficients and a 0 for the fund indicator
    obj = np.array(simmat.flatten().tolist()+[0]*len(allstocks))


    A = np.zeros((np.shape(obj)[0]+1,np.shape(obj)[0]))
    b=np.zeros(np.shape(A)[0])
    s = np.array(['']*np.shape(A)[0])

    #Each stock represented
    ind_vec = np.array(range(len(allstocks)))
    row = 0
    for j in range(len(allstocks)):
        A[row,j*len(allstocks) + ind_vec] = 1
        b[row] = 1
        s[row] = '='
        row+=1

    #best representative identification
    for j in range(len(allstocks)):
        xind = j
        yind = xind + len(allstocks)**2
        for i in range(len(allstocks)):
            A[row,[xind,yind]] = [1,-1]
            b[row] = 0
            s[row] = '<'
            xind+=100
            row+=1


    #Fund Size
    A[row,-len(allstocks):] = [1]*len(allstocks)
    b[row] = m
    s[row] = '='
    len(s)

    #Optimize
    lpMod = gp.Model()
    lpMod_x = lpMod.addMVar(len(obj), vtype = 'B')
    lpMod_con = lpMod.addMConstrs(A, lpMod_x, s, b)
    lpMod.setMObjective(None,obj,0,sense=gp.GRB.MAXIMIZE)
```

```
    lpMod.Params.OutputFlag = 0 # tell gurobi to shut up!!
    lpMod.optimize()

    return lpMod.x[len(allstocks)**2:]
```

After that we defined a function to calculate the portfolio weights:

```
def getweights(fund_indexes,s2019,returns,allstocks,fund_names,l=''):
    if l =='l':
        return -1
    #Get Index Returns
    ndxprices = s2019.NDX
    i=0
    ndxret = []
    while i<len(ndxprices)-1:
        cur = ndxprices[i]
        nex = ndxprices[i+1]
        ndxret.append((nex-cur)/cur)
        i+=1
    #Get relevant fund returns
    fundreturns = pd.DataFrame(data = ndxret,columns=['NDX'])
    c=0
    for s in fund_indexes:
        if s == 1:
            temp = returns.iloc[:,c]
            fundreturns[allstocks[c]] = temp
        c+=1


    #[w1,w2,w3,w4,w5,d1...d250]
    obj = np.ones((len(fundreturns)+len(fund_names)))
    obj[:len(fund_names)] = 0
    #A[501,350]
    A = np.zeros((1+(len(fundreturns)*2),len(fundreturns)+len(fund_names)))
    b=np.zeros((1+(len(fundreturns)*2)))
    s = np.array([''](1+(len(fundreturns)*2)))

    row = 0
    A[row,:len(fund_names)] = [1]
    b[row] = 1
    s[row] = '='
    row+=1
```

```
    counter = len(fund_names)
    c = 0
    while row < A.shape[0]:
        returns_t = list(fundreturns.iloc[c,1:].values)
        neg_returns_t = [ -x for x in returns_t]
        #print('Const:', row)
        #print('Returns:', returns_t)
        #print('d index:',counter)
        A[row,0:len(fund_names)] = returns_t
        A[row,counter] = 1
        b[row] = fundreturns.iloc[c,0]
        s[row] = '>'
        row+=1
        #print('Const:', row)
        #print('Negative:', neg_returns_t)
        #print('d index:',counter)
        A[row,0:len(fund_names)] = neg_returns_t
        A[row,counter] = 1
        b[row] = -1*(fundreturns.iloc[c,0])
        s[row] = '>'
        counter+=1
        c+=1
        row+=1



    lpMod = gp.Model()
    lpMod_x = lpMod.addMVar(len(obj))
    lpMod_con = lpMod.addMConstrs(A, lpMod_x, s, b)
    lpMod.setMObjective(None,obj,0,sense=gp.GRB.MINIMIZE)

    lpMod.Params.OutputFlag = 0 # tell gurobi to shut up!!
    lpMod.optimize()

    return lpMod.x[:len(fund_names)]
```

Indirect Method: Here, we completely ignored the stock selection IP and re-formulated the weight selection problem to be an MIP that constrains the number of non-zero weights to be an integer. To do this, we took the weight selection problem and replaced m with n so that we optimize over all weights subject to the following constraints:

$$min \sum_{t=1}^{T} y_t$$

$$\text{s.t.} \sum_{i=1}^{m} w_i = 1$$

$$\sum_{i=1}^{n} y_i = m$$

$$w_i - My_i <= 0 \quad for\ i = 1\ to\ n$$

$$y_t + \sum_{i=1}^{n} w_i r_{it} >= q_t \quad for\ t = 1\ to\ T$$

$$y_t - \sum_{i=1}^{n} w_i r_{it} >= -q_t \quad for\ t = 1\ to\ T$$

$$w_i >= 0$$

Python Code:

We defined the function to calculate portfolio weights directly (without stock selection):

```
def getweightsfund(fund_indexes,s2019,returns,allstocks,fund_names,m,l=''):
    if l =='l':
        return -1
    #Get NASDAQ Returns
    ndxprices = s2019.NDX
    i=0
    ndxret = []
    while i<len(ndxprices)-1:
        cur = ndxprices[i]
        nex = ndxprices[i+1]
        ndxret.append((nex-cur)/cur)
        i+=1
    fundreturns = pd.DataFrame(data = ndxret,columns=['NDX'])
    #fundreturns

    c=0
    for s in fund_indexes:
        if s == 1:
            temp = returns.iloc[:,c]
            fundreturns[allstocks[c]] = temp
        c+=1

    #[w1,w2,w3,w4,w5,d1...d250]
    #obj = np.array()
    obj = np.zeros((len(fundreturns)+len(fund_names)*2))
    obj[len(fund_names):-len(fund_names)] = 1
    vtype = np.array(['C']*(len(fundreturns)+len(fund_names))+['B']*len(fund_names))
    #print(obj)

    A = np.zeros((2+(len(fundreturns)*2)+len(fund_names),len(fundreturns)+len(fund_names)*2))
    b = np.zeros((2+(len(fundreturns)*2)+len(fund_names)))
    s = np.array(['']*(2+(len(fundreturns)*2)+len(fund_names)))


    row = 0
    A[row,:len(fund_names)] = [1]
    b[row] = 1
    s[row] = '='
    row+=1
```

```
counter = len(fund_names)
c = 0
while row < A.shape[0]-len(fund_names)-1:
    returns_t = list(fundreturns.iloc[c,1:].values)
    neg_returns_t = [ -x for x in returns_t]

    A[row,0:len(fund_names)] = returns_t
    A[row,counter] = 1
    b[row] = fundreturns.iloc[c,0]
    s[row] = '>'

    row+=1

    A[row,0:len(fund_names)] = neg_returns_t
    A[row,counter] = 1
    b[row] = -1*(fundreturns.iloc[c,0])
    s[row] = '>'

    counter+=1
    c+=1
    row+=1
counter = 0

while row < A.shape[0]-1:
    A[row,[counter,counter+len(fund_names)+len(fundreturns)]] = [1,-1]
    b[row] = 0
    s[row] = '<'
    counter+=1
    row+=1

A[row,-len(fund_names):] = 1
b[row] = m
s[row] = '='
print('Optimizing Level: {}'.format(m))
lpMod = gp.Model()
lpMod.setParam('TimeLimit', timeout)
lpMod_x = lpMod.addMVar(len(obj),vtype = vtype)
lpMod_con = lpMod.addMConstrs(A, lpMod_x, s, b)
lpMod.setMObjective(None,obj,0,sense=gp.GRB.MINIMIZE)

lpMod.Params.OutputFlag = 0 # tell gurobi to shut up!!
lpMod.optimize()

mod_optimalval =  lpMod.Objval
mod_descisions = lpMod.x[:len(fund_names)]
print('Done Optimizing')
return mod_optimalval, mod_descisions
```

Performance metric: we utilized a performance metric to compare the two models and their in and out-sample performances. We used the performance score defined as the aggregate absolute difference between the index and portfolio returns over a period for different #stocks.

Python Code:
We defined the function to compute the portfolio performance score:

```
def getaccuracy(fund_names,returns2020, weights_l,l=''):
    if l =='l':
        return -1
    fund = returns2020[fund_names+['NDX']]

    weights = pd.Series(data = weights_l)

    weighted_returns= fund.copy()#.astype('float')
    for i in range(len(weights_l)):
        weighted_returns.iloc[:,i] = weighted_returns.iloc[:,i]*weights_l[i]

    weighted_returns['pred'] = weighted_returns.iloc[:,:-1].apply(np.sum, axis=1)
    weighted_returns['dif'] = abs(weighted_returns.iloc[:,-2]-weighted_returns.iloc[:,-1])
    return sum(weighted_returns['dif']), list(weighted_returns['pred'])
```

Insights and Recommendation

The below tables give us the Out-Sample Performance Scores for the two models across different #stocks.

#3. Differences:

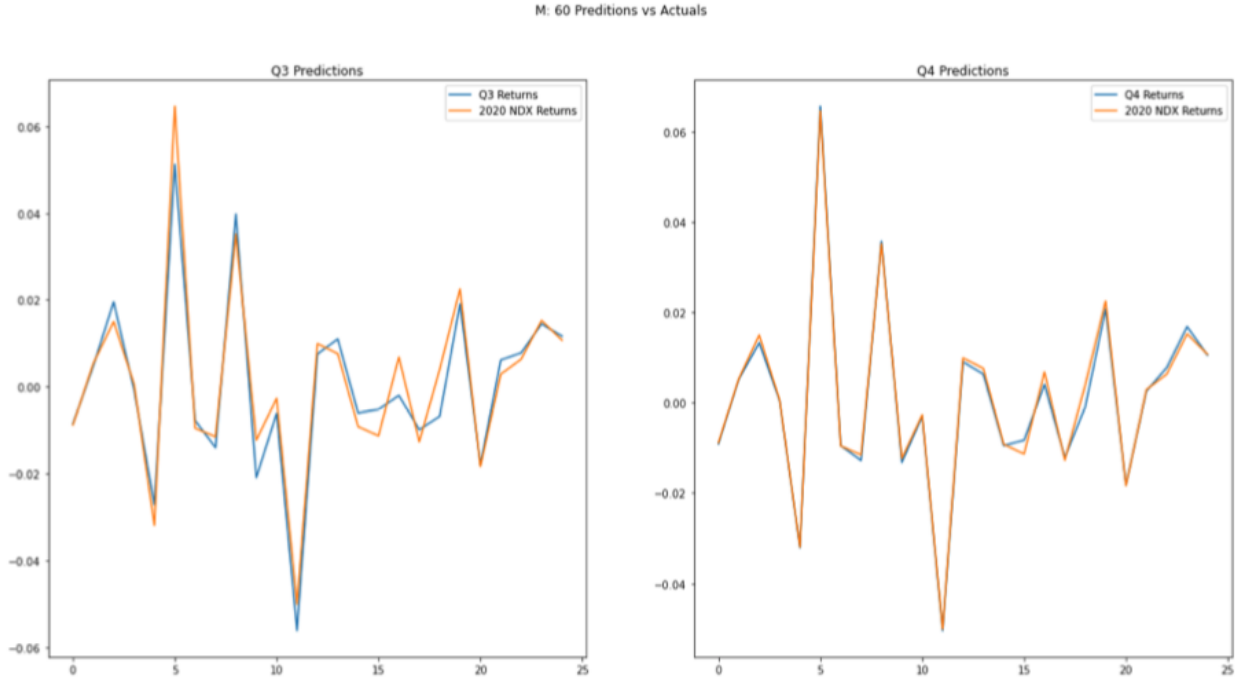
	Difference
5	1.109405
10	1.096172
20	0.897668
30	0.755453
40	0.819101
50	0.771502
60	1.164219
70	0.832701
80	0.543650
90	0.364608
100	0.365491

#4. Complex Differences:

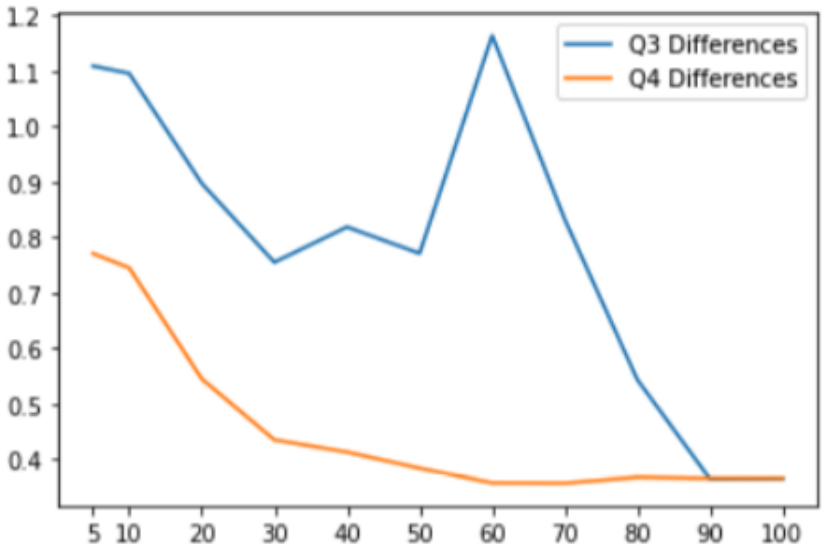
	Difference
5	0.771303
10	0.745532
20	0.545204
30	0.435238
40	0.413289
50	0.384137
60	0.357563
70	0.357104
80	0.367422
90	0.365491
100	0.365491

The below plot shows the Fund vs. Index Returns (Indirect (Q3) vs. Direct Method (Q4)) for m=60.

```
m: 60
```



The below plot shows the In and Out-Sample Performance Scores (Indirect vs. Direct Method).



In-Sample vs Out-Sample Performance (Indirect Method)

The portfolio model tends to perform relatively poorly for the out-sample 2020 data vs. the in-sample 2019 data. The 2020 line graph, which represents the difference in the performance between the index and our fund (performance score), is above the 2019 one for all counts of stocks. Also, the decline in the error is much smoother in 2019, while we see a sharp jump in the error for the year 2020 when the number of stocks nears 60 (Q3 Differences) and then a sharper decline after that.

Fund Performance vs. Number of stocks

Both Out-Sample plots are downward sloping indicating that as the number of stocks increase, the difference between the index return and the fund return decreases and hence it's advantageous to keep increasing the number of stocks until a point where further increase in the number of stocks does not decrease the performance difference by much i.e. there are diminishing returns from including more stocks in the portfolio. We see this happen when the number of stocks reach around 90 for the Indirect Model (Q3 Difference) and around 60 for the Direct Model (Q4 Difference).

Direct vs. Indirect Method

The table and the graphs above show that the Direct Method (Q4) performs better on the 2020 data than the original Indirect Method (Q3) across all numbers of stocks. The difference between index and the fund returns (i.e. performance score) are significantly lower for the Direct Method for all counts of stocks, meaning that this method does a better job tracking the index vs. the Indirect Method.

Therefore, we would recommend to use the Direct Method to form the index fund in order to minimize the performance error.